> This is a direct transition from the specification to code

> What about (graphical) modeling?

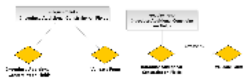
> May be required by the project, help communicate design decisions, or improve reuse, especially in model-driven settings

How to model changes for their aspect-oriented realization ?

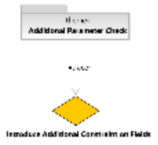
Identify the themes in the change request

Determine the constraining theme

Find the corresponding specification change type in the catalog

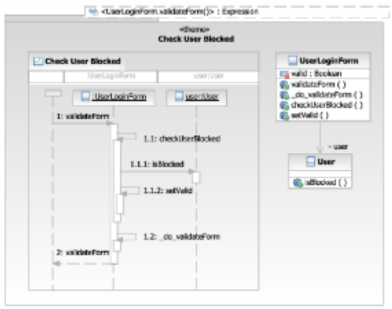


Find the matching realization change type in the catalog

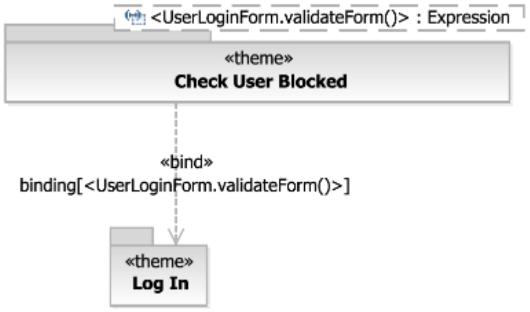

Additional Parameter Check
Introduce Additional Constraint on Fields

Additional Parameter Check

Use the matching realization change type as a template for the change to be designed



Apply the change to the original model element


<UserLoginForm.validateForm()> : Expression
«theme»
Check User Blocked
«bind»
binding[<UserLoginForm.validateForm()>]
«theme»
Log In

> Within the common (non aspect-oriented modeling) we can expect that the larger the change is, more diagrams we have to edit

> In our approach, the change is focused in one diagram and the original design is not being affected at all

Realizing Changes by Aspects at the Design Level

Valentino Vranić and Branislav Kuliha

Institute of Informatics and Software Engineering

STU FIIT

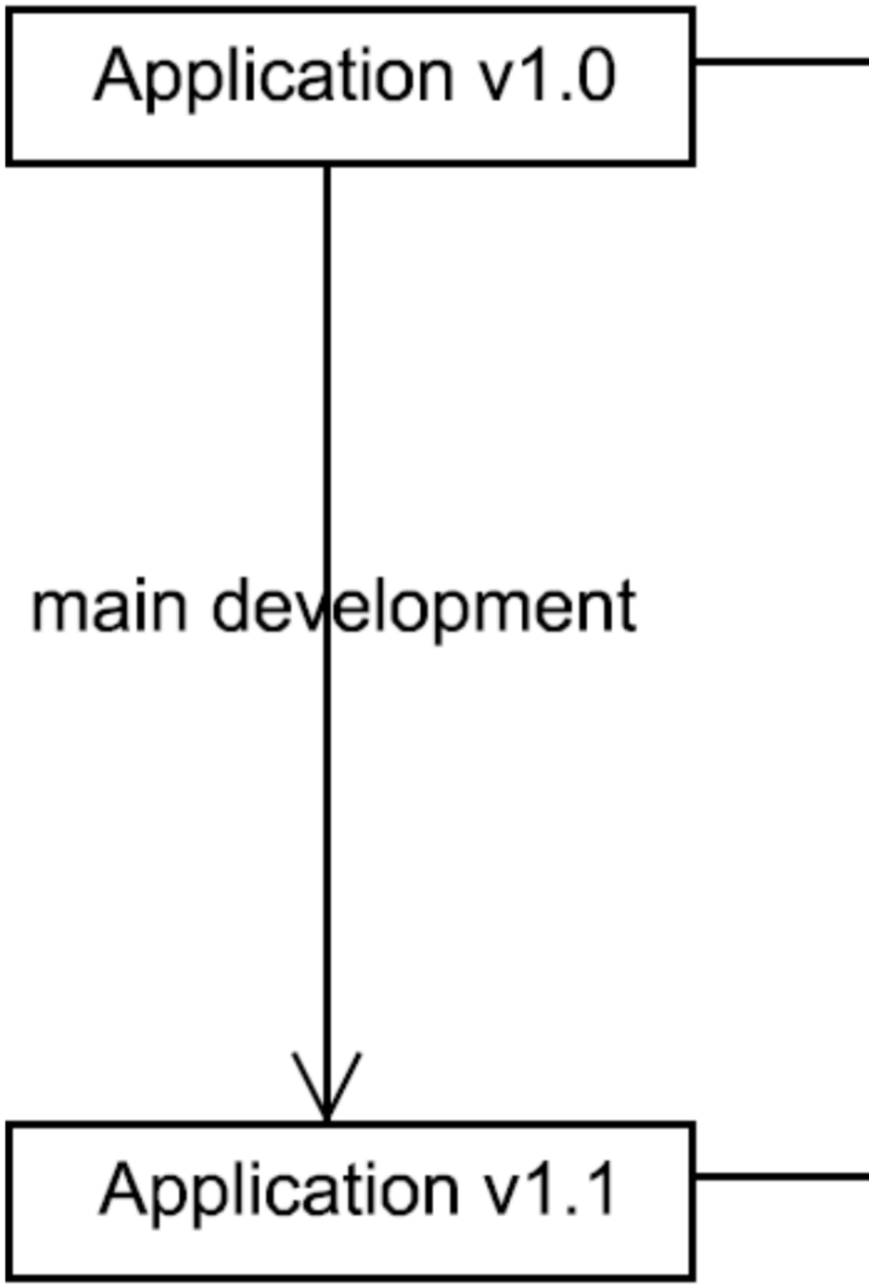
SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

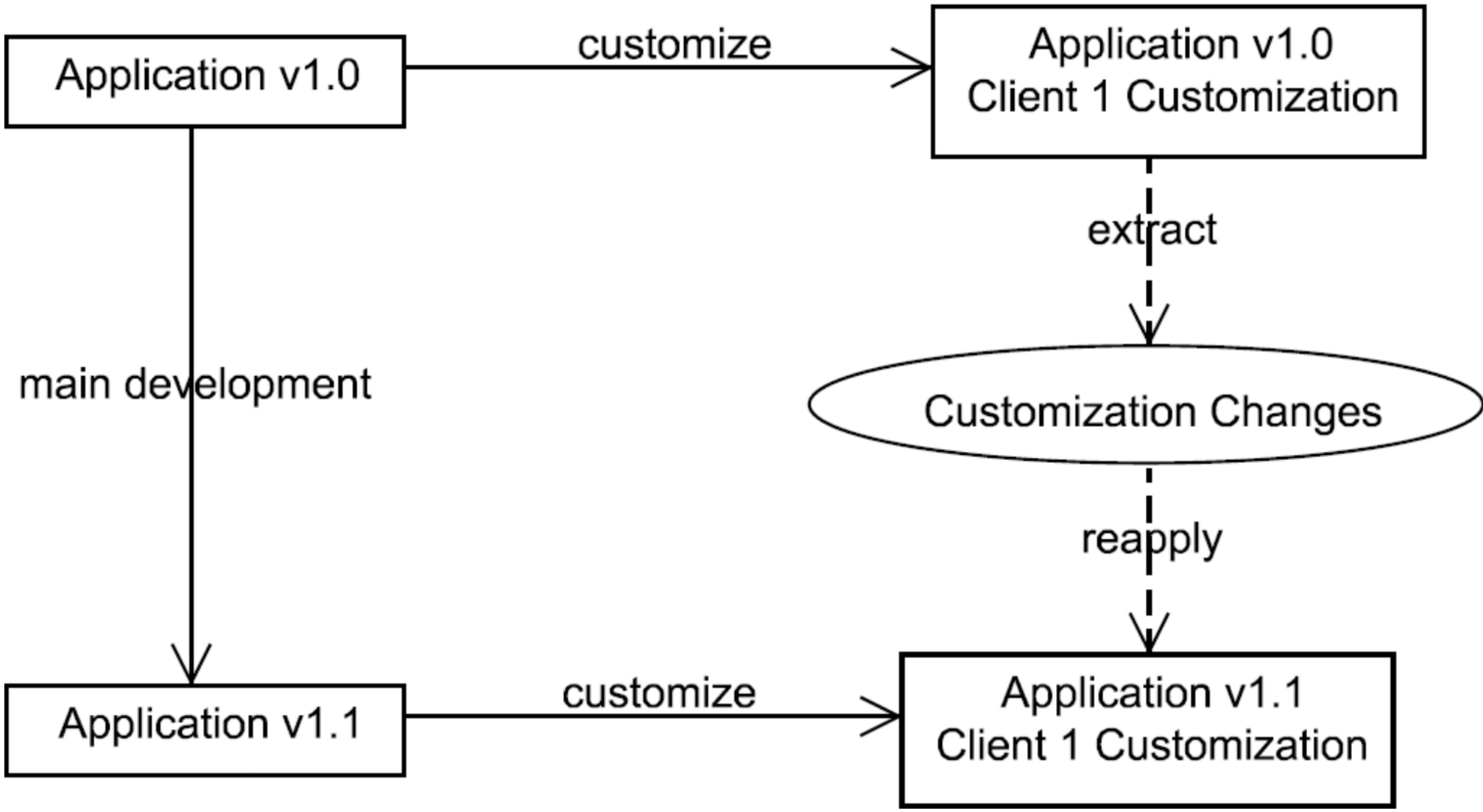
vranic@stuba.sk
fiit.sk/~vranic

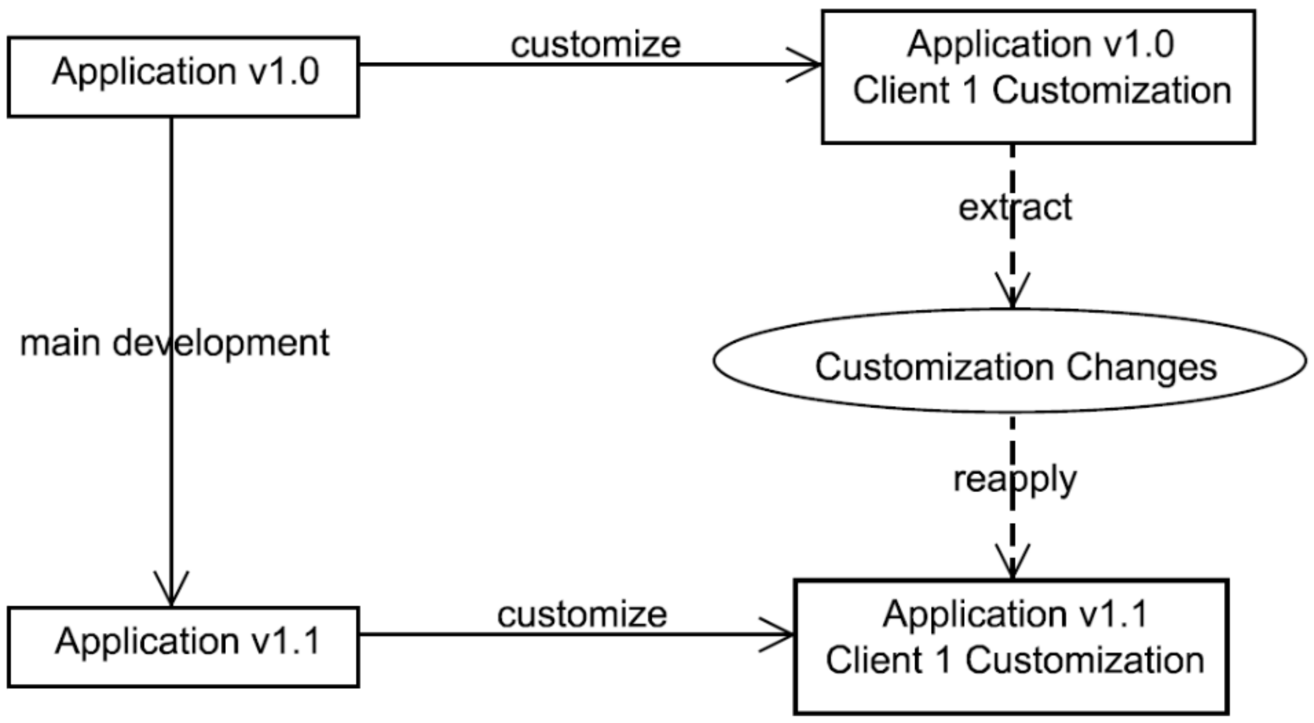
kobliha@centrum.sk

INES 2015

Bratislava, September 5, 2015


Application v1.0
main development
Application v1.1


Application v1.0
customize
Application v1.0 Client 1 Customization
extract
main development
Customization Changes
reapply
Application v1.1
customize
Application v1.1 Client 1 Customization


Application v1.0
customize
Application v1.0 Client 1 Customization
extract
main development
Customization Changes
reapply
Application v1.1
customize
Application v1.1 Client 1 Customization

Aspects

Aspect-oriented programming, AspectJ...

```
public class Point {
    private int x;
    private int y;

    public void setX(int x) { this.x = x; }
    public void setY(int y) { this.y = y; }
    public int getX() { return x; }
    public int getY() { return y; }
}
```

```
public aspect AccessMonitoring {
    before(): execution(void Point.set*(..)) {
        System.out.println("Moving a point.");
    }
    before(): execution(int Point.get*(..)) {
        System.out.println("Reading a point.");
    }
    after(): execution(* Point.set*(..)) {
        System.out.println("Moved a point.");
    }
    after(): execution(* Point.get*(..)) {
        System.out.println("Read a point.");
    }
}
```

Aspect-oriented programming, AspectJ...

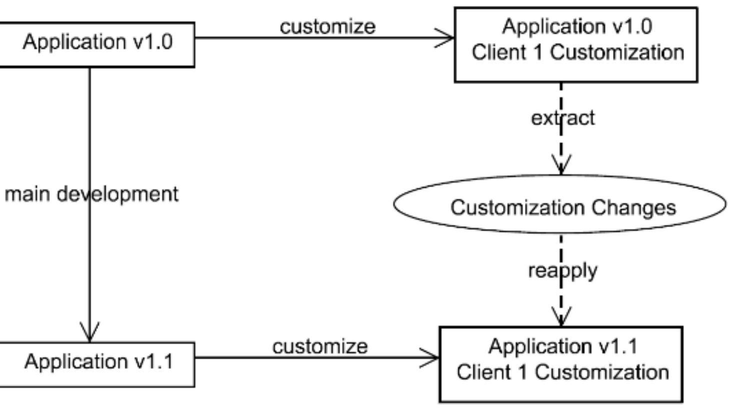
```
public class Point {
    private int x;
    private int y;

    public void setX(int x) { this.x = x; }
    public void setY(int y) { this.y = y; }
    public int getX() { return x; }
    public int getY() { return y; }
}
```

```
public aspect AccessMonitoring {
    before(): execution(void Point.set*(..)) {
        System.out.println("Moving a point.");
    }
    before(): execution(int Point.get*(..)) {
        System.out.println("Reading a point.");
    }
    after(): execution(* Point.set*(..)) {
        System.out.println("Moved a point.");
    }
    after(): execution(* Point.get*(..)) {
        System.out.println("Read a point.");
    }
}
```

```
public aspect RangeControl {
    void around(int x): call(void Point.setX(..)) && args(x) {
        if (x < 0)
            proceed(640 + x % 640);
        else if (x > 639)
            proceed(x % 640);
        else
            proceed(x);
    }

    void around(int y): call(void Point.setY(..)) && args(y) {
        if (y < 0)
            proceed(400 + y % 400);
        else if (y > 400)
            proceed(y % 400);
        else
            proceed(y);
    }
}
```



Aspects

Aspect-oriented change realization

CHR03:

The administrator should be able to block and unblock an account from the accounts view.

Change CHR03-1: The administrator can block and unblock an account from the accounts view

Change CHR03-2: A user cannot log in if his/her account is blocked

Change CHR03-1: The administrator can block and unblock an account from the accounts view

→ Aspects

Change CHR03-2: A user cannot log in if his/her account is blocked

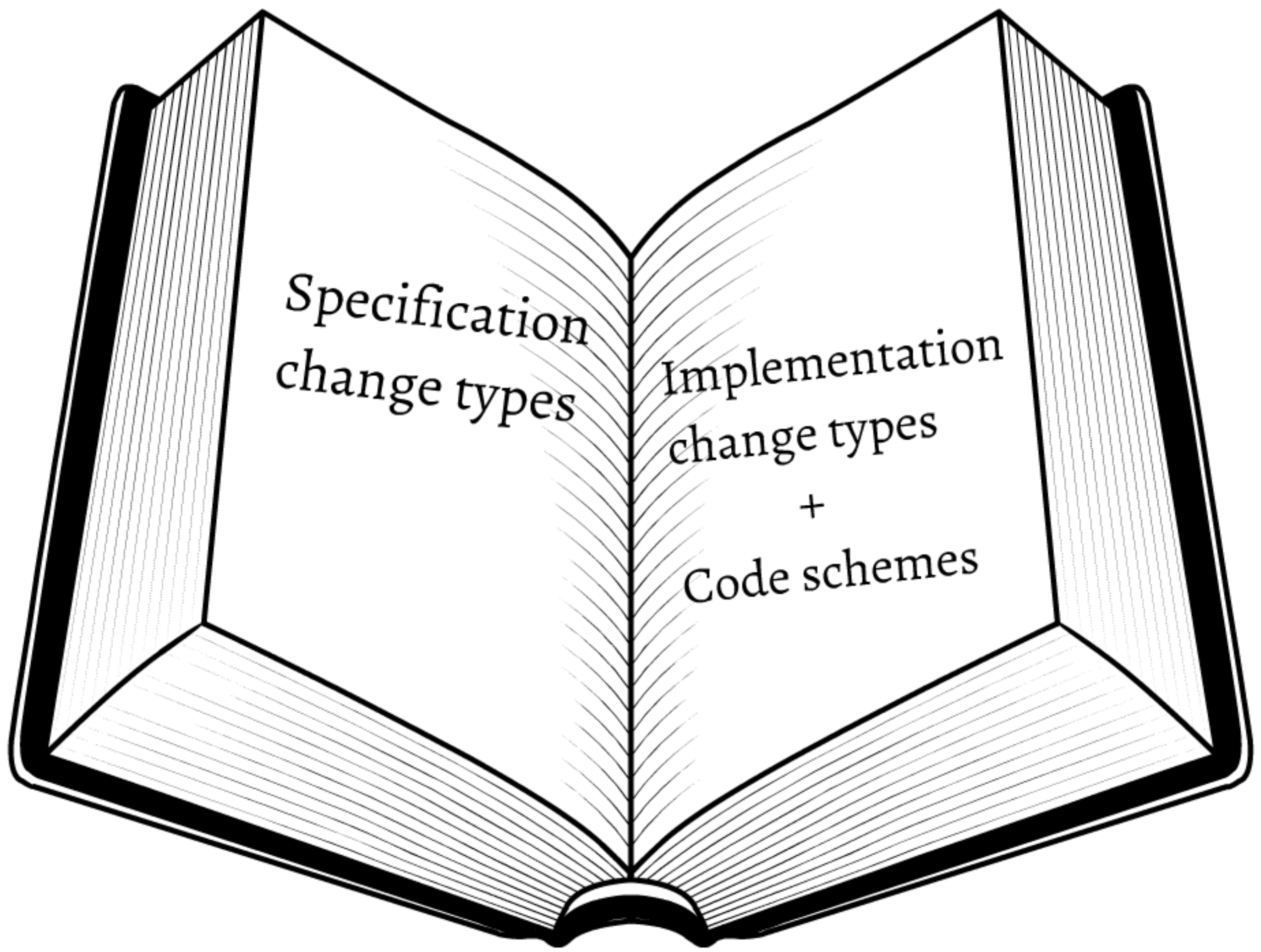
But how to make them ?

> Different changes share essential properties forming change types

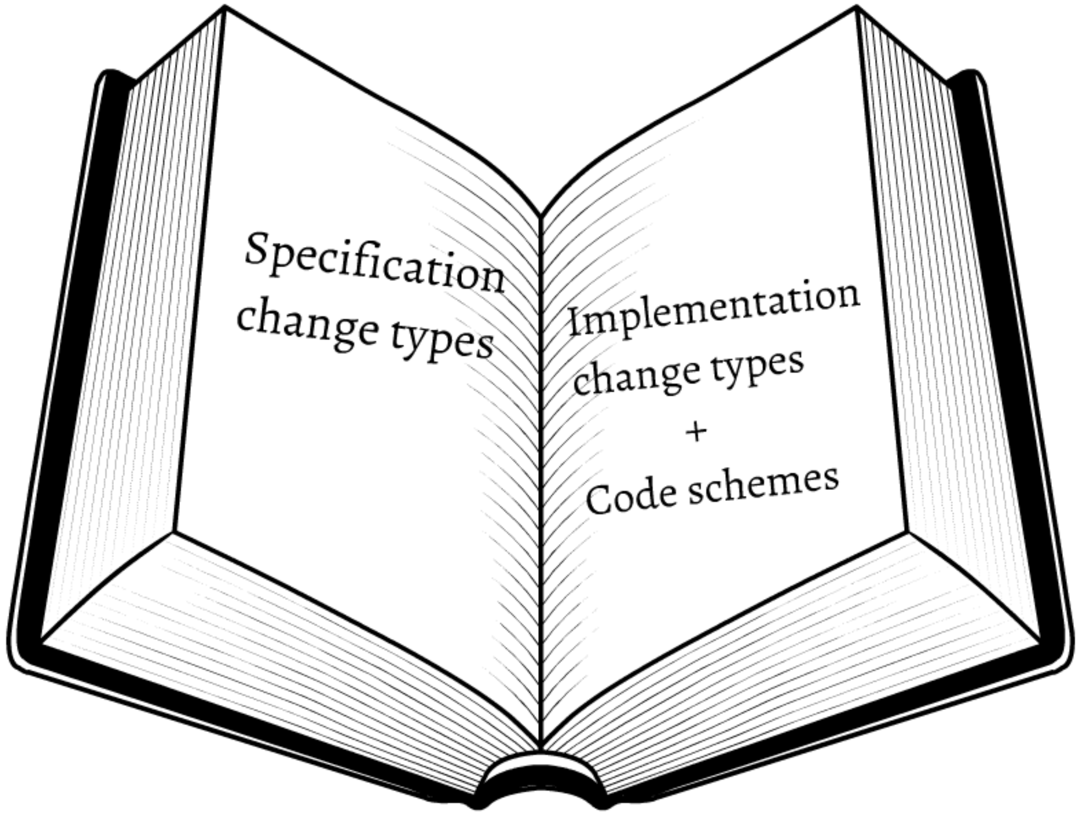
> This happens both at the specification and implementation level

> Different changes share essential properties forming change types

> This happens both at the specification and implementation level


Specification change types
Implementation change types + Code schemes

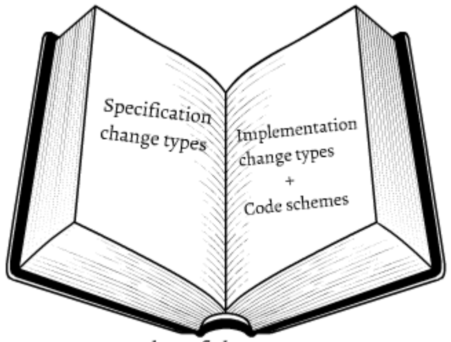
Catalog of change types



Catalog of change types

1. Generalize the change (description)
2. Find the corresponding specification change type in the catalog
3. Apply the matching implementation type with its code scheme

implementation level


Specification change types
Implementation change types + Code schemes

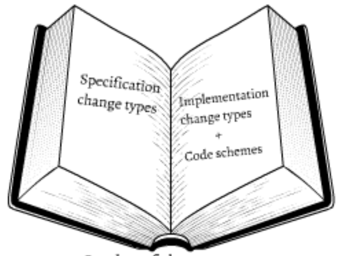
Catalog of change types

1. Generalize the change (description)
2. Find the corresponding specification change type in the catalog
3. Apply the matching implementation type with its code scheme

> This is a direct transition from the specification to code

> What about (graphical) modeling?

> May be required by the project, help communicate design decisions, or improve reuse, especially in model-driven settings



Catalog of change types

1. Generalize the change (description)
2. Find the corresponding specification change type in the catalog
3. Apply the matching implementation type with its code scheme

> This is a direct transition from the specification to code

> What about (graphical) modeling?

> May be required by the project, help communicate design decisions, or improve reuse, especially in model-driven settings

How to model changes for their aspect-oriented realization ?

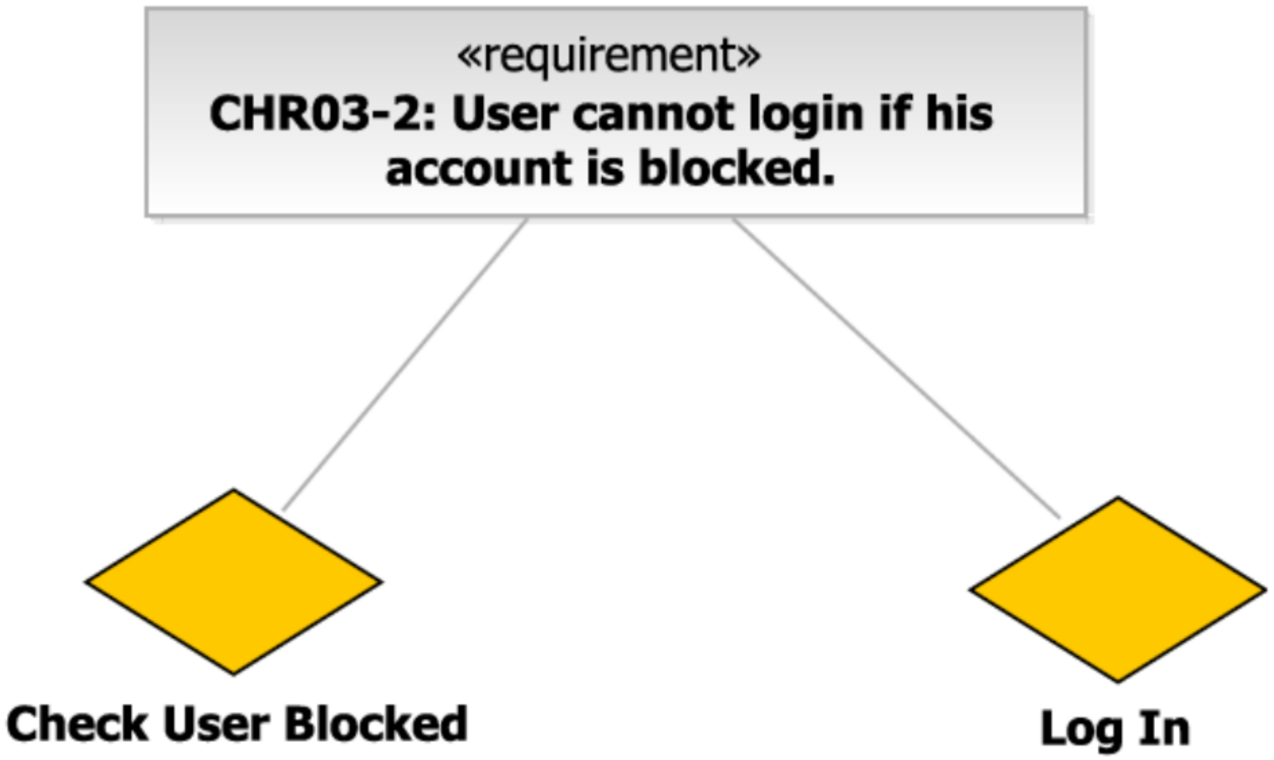
> No industry accepted approach to aspect-oriented modeling

> We used Theme: software models expressed in terms of so-called *themes*, i.e., concerns

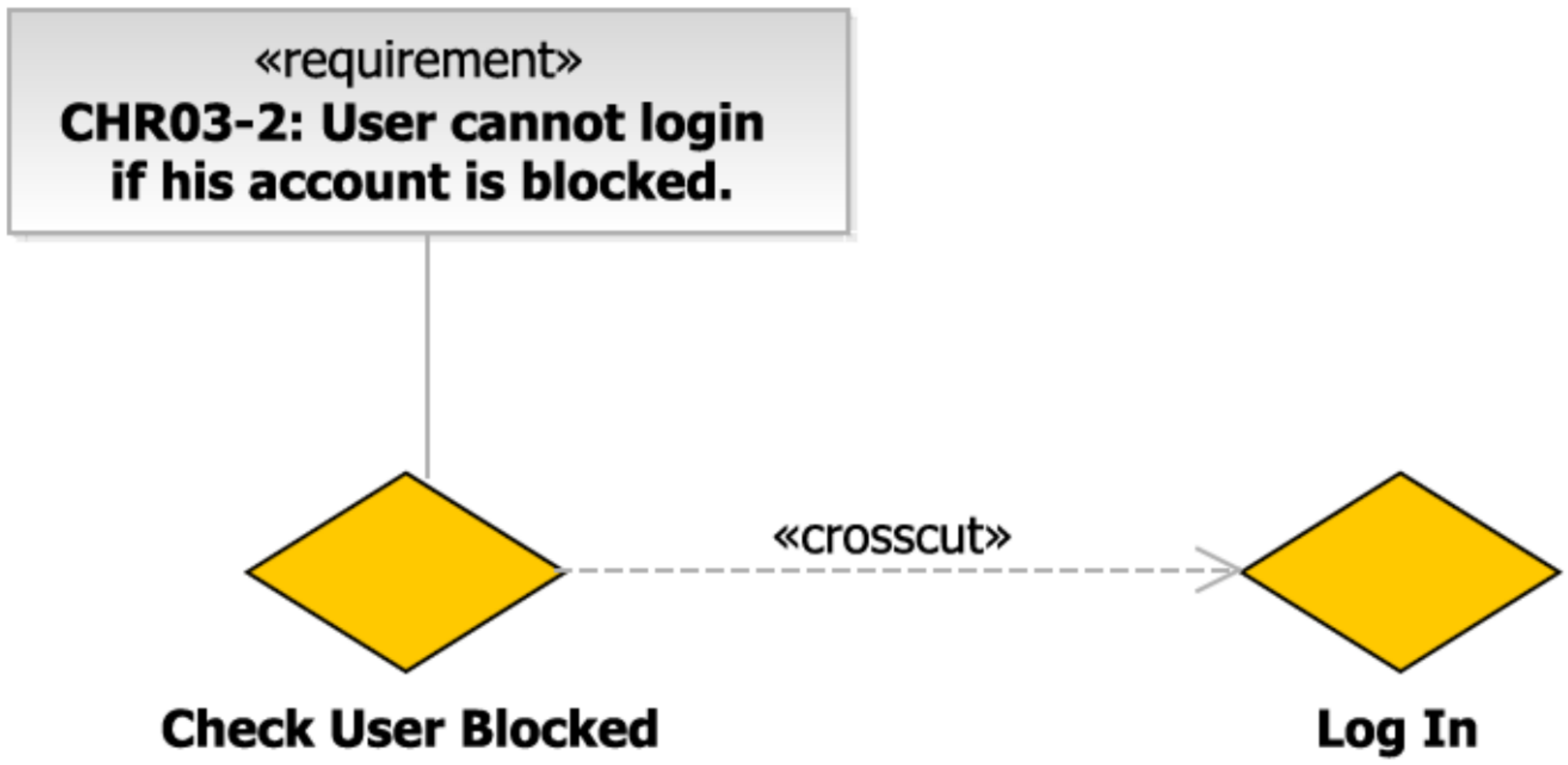
> Theme/Doc: specification/analysis (requirements and themes)

> Theme/UML: design (themes as parameterized packages)

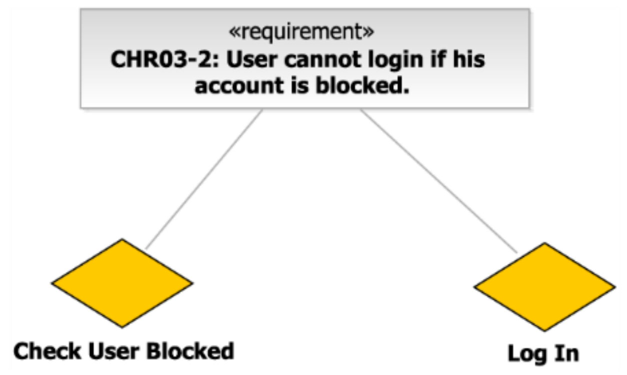
Identify the themes in the change request



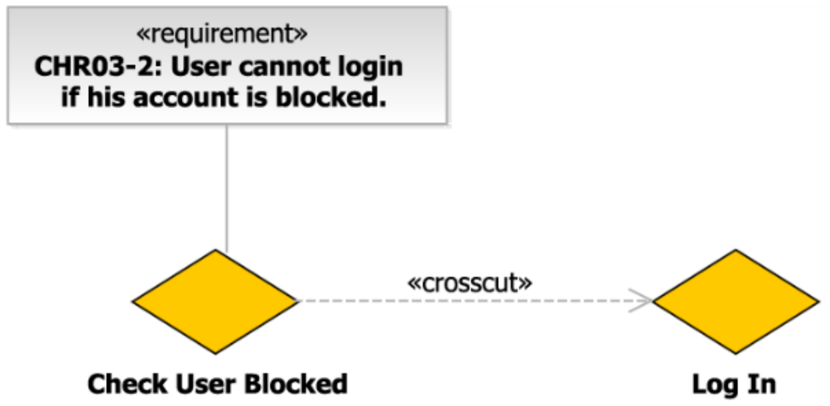
Determine the crosscutting theme



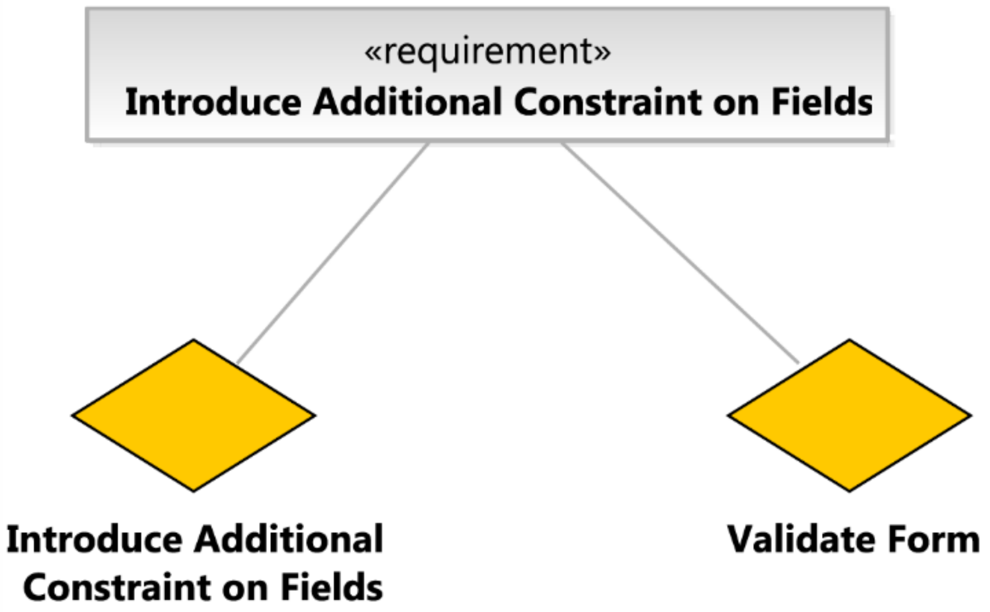
Identify the themes in the change request

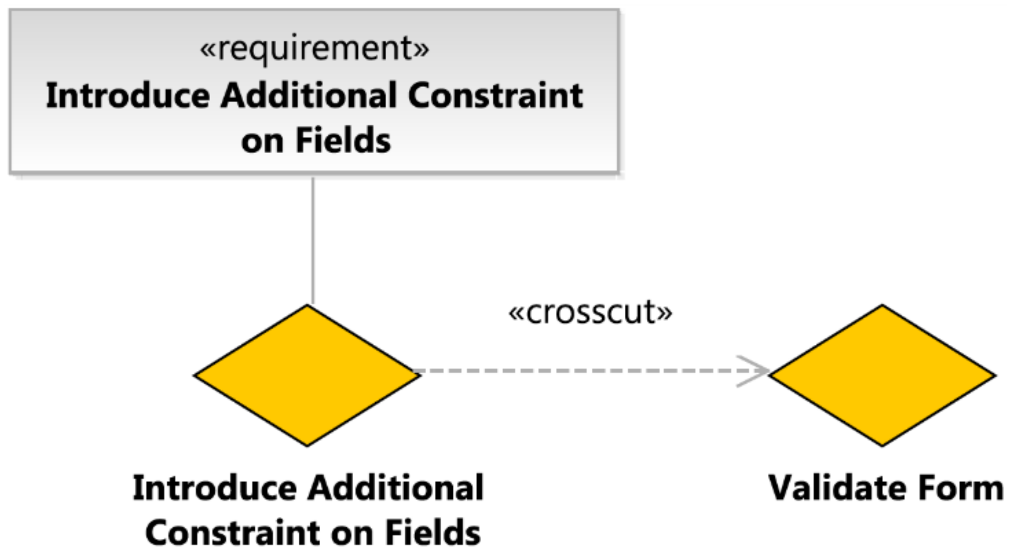


Determine the crosscutting theme

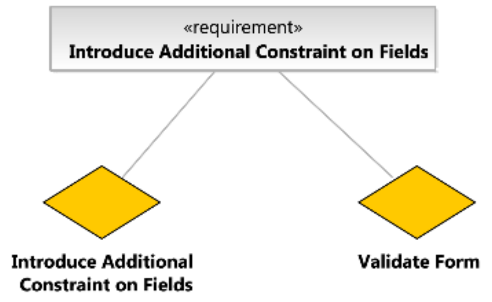


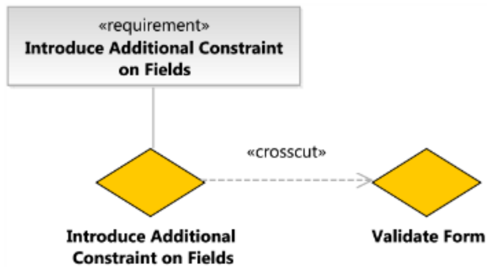
Find the corresponding specification change type in the catalog


«requirement»
Introduce Additional Constraint on Fields
Introduce Additional Constraint on Fields
Validate Form

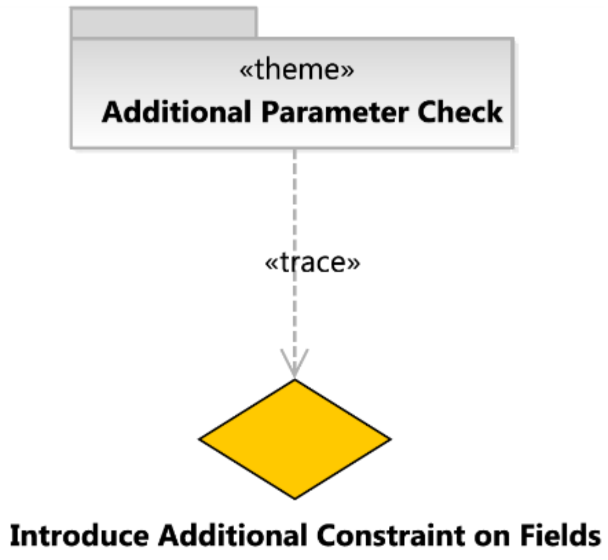

«requirement»
Introduce Additional Constraint on Fields
«crosscut»
Introduce Additional Constraint on Fields
Validate Form

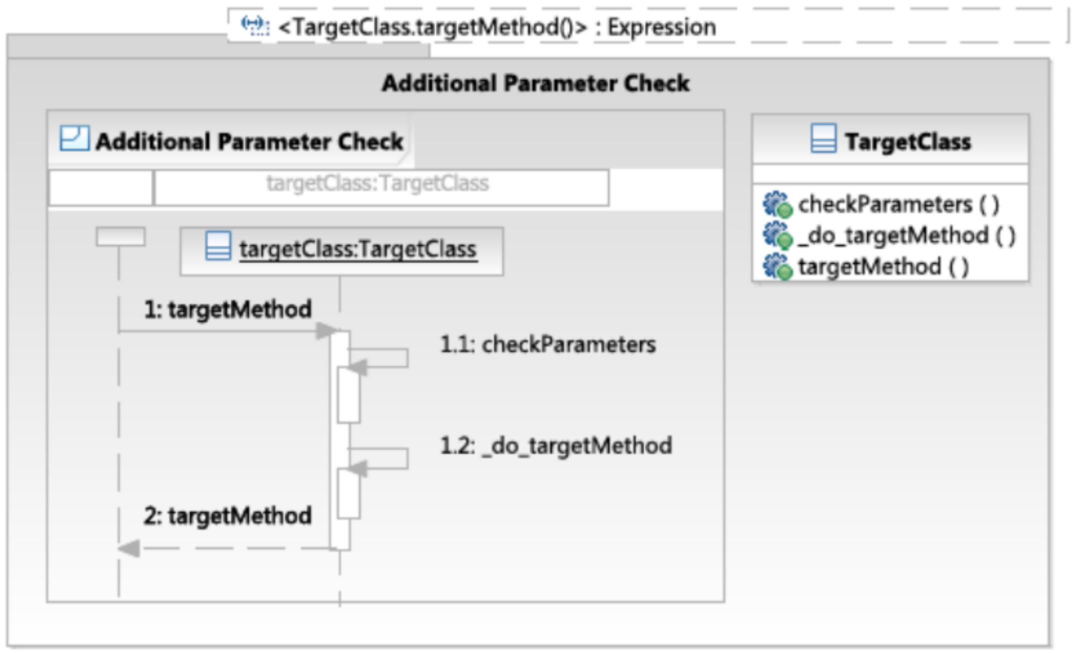
Find the corresponding specification change type in the catalog

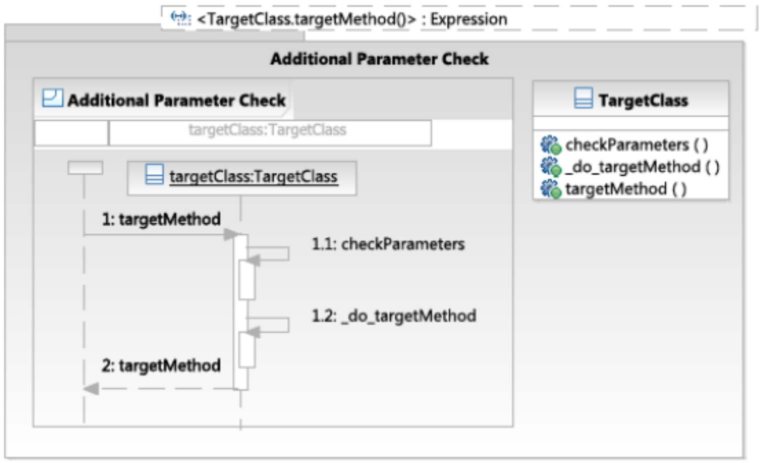

«requirement»
Introduce Additional Constraint on Fields
Introduce Additional Constraint on Fields
Validate Form


«requirement»
Introduce Additional Constraint on Fields
«crosscut»
Introduce Additional Constraint on Fields
Validate Form

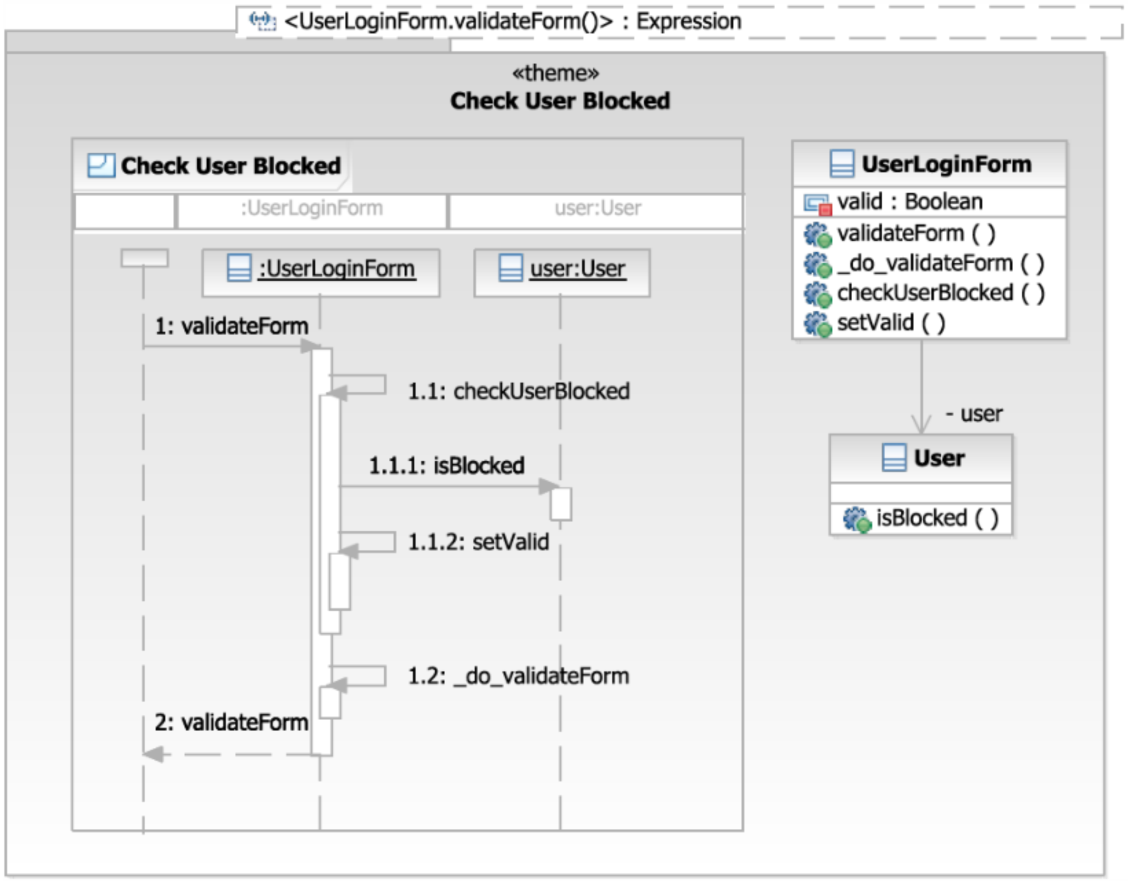
Find the matching realization change type in the catalog


«theme»
Additional Parameter Check
«trace»
Introduce Additional Constraint on Fields

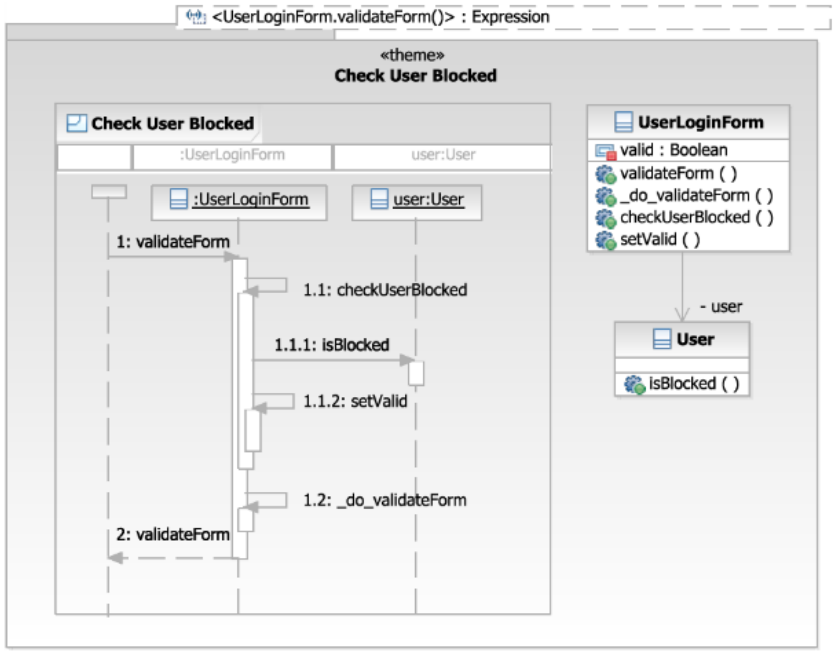




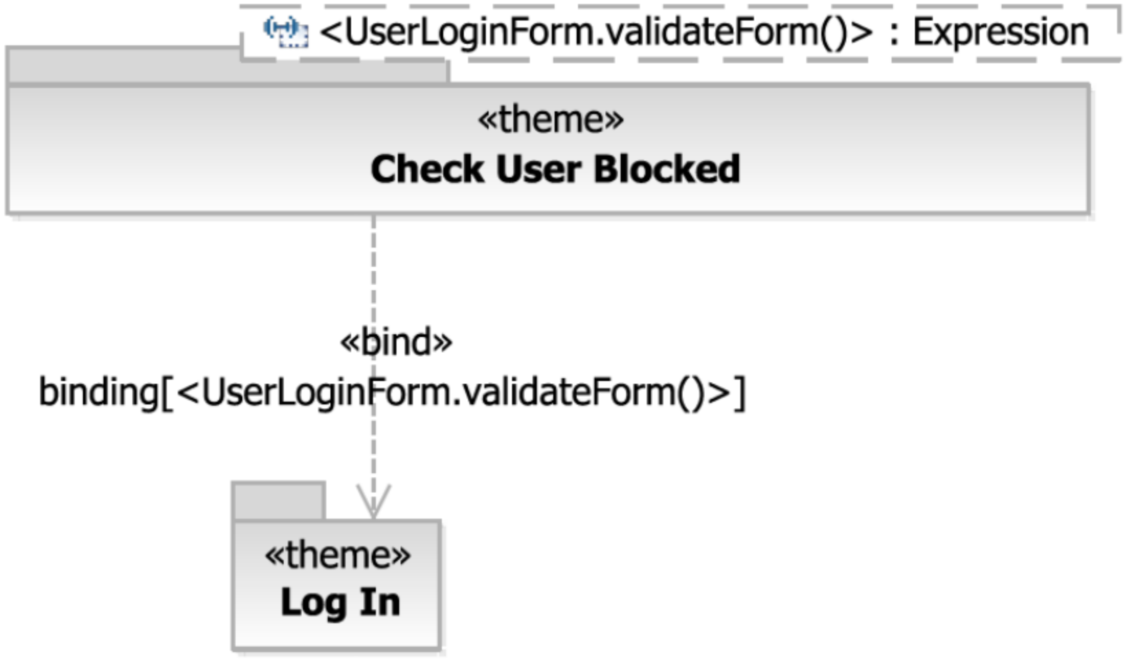
Use the matching realization change type as a template for the change to be designed



Use the matching realization change type as a template for the change to be designed

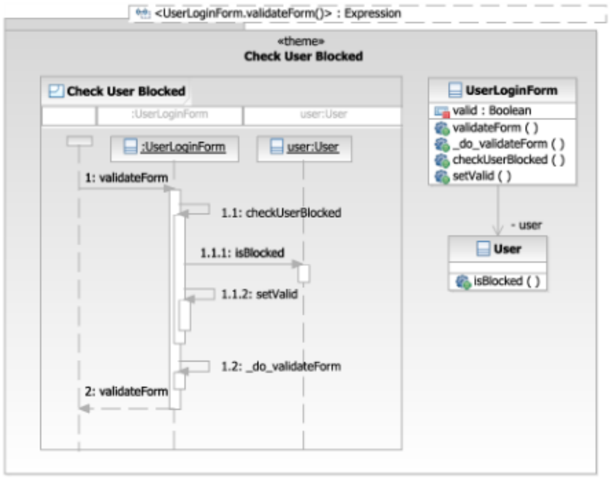


Apply the change to the original model element

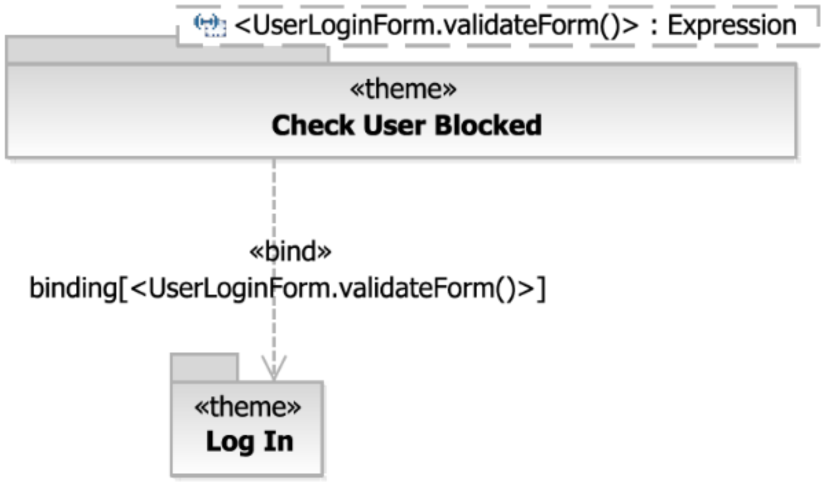


<UserLoginForm.validateForm()> : Expression
«theme»
Check User Blocked
«bind»
binding[<UserLoginForm.validateForm()>]
«theme»
Log In

Use the matching realization change type as a template for the change to be designed



Apply the change to the original model element


<UserLoginForm.validateForm()> : Expression
«theme»
Check User Blocked
«bind»
binding[<UserLoginForm.validateForm()>]
«theme»
Log In

> Within the common (non aspect-oriented modeling) we can expect that the larger the change is, more diagrams we have to edit

> In our approach, the change is focused in one diagram and the original design is not being affected at all

Summary

> All change types from the existing catalog for the domain of web applications have been modeled: 11 specification change types and 7 implementation change types

> A UML profile for Theme/Doc and Theme/UML has been designed and implemented in IBM Rational Software Architect

> The evaluation was conducted on a real web mail system

> Two change requests for the web mail system were studied, analyzed, designed, and implemented

> The variants of change types should be recorded, including the implementation technology

> The feedback from the developers would help improve the catalog

vranic@stuba.sk
fiit.sk/~vranic